EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	1	((embed$5 encapsulat$5) same ((user and kernel) and ((forc$3 invok$4) near8 exception))).clm.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/24 15:51
S1	4	(@ad<="20011115" or @rlad<="20011115") and (((kernel and user) adj2 level) same debug$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/24 14:06

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S120	2	"5522075".pn.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/14 16:14
S121	113	(kernel with (execution executed code)) same (interrupt near2 handler)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/14 16:15
S122	12	S121 and (handler same (embedded encapsulat$4 instrumentation inlin$3))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/14 16:17
S123	23	S121 not S122 and (((user protected) and (unprotected kernel)) near mode)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/14 16:18
S124	12	(@ad<="20011115" or @rlad<="20011115") and S123	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/14 16:25
S125	0	S124 and (instrumentation inline)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/14 16:25
S126	0	S124 and (instrumentation in-line)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/14 16:25
S127	0	S124 and (instrumented in-line inlin$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/14 16:25

EAST Search History

S128	36	(US-20020131404-$ or US-20020129337-$ or US-20020073402-$ or US-20020087883-$ or US-20040040013-$ or US-20010005852-$ or US-20020120854-$).did. or (US-6708326-$ or US-6438666-$ or US-6385727-$ or US-5963737-$ or US-6074427-$ or US-6543049-$ or US-5630049-$ or US-5774724-$ or US-6219828-$ or US-5812850-$ or US-6047124-$ or US-6378125-$ or US-6305010-$ or US-6158045-$ or US-5889988-$ or US-5680584-$ or US-6480818-$ or US-6754851-$ or US-5463764-$ or US-6895576-$ or US-6732296-$ or US-5450586-$ or US-6353923-$ or US-6859891-$ or US-6077312-$ or US-6760907-$). did. or (US-4910663-$ or US-6842893-$).did. or (EP-695994-$).did.	US-PGPUB; USPAT; DERWENT	OR	OFF	2007/02/15 08:22
S129	127	(anti-piracy obfuscat$4 encapsulat$4 debug$4) and ((user protected) adj2 mode) and ((kernel unprotected) adj2 mode) and (interrupt near5 (trigger$4 invok$3 handl$4 breakpoint))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/15 08:27
S130	66	((user mode) near3 protected) same (((invok$4 call execut$4) with interrupt) same handler)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/15 08:27
S131	0	S129 and S130	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/02/15 08:27
S132	10	S130 and ((user protected) adj2 mode) and ((kernel unprotected) adj2 mode) and (interrupt near5 (trigger$4 invok$3 handl$4 breakpoint))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/15 09:50
S133	0	(((in-line inlin$4) with (code instruction command macro directive operation)) same (interrupt with handl$4)).clm.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/15 09:52

EAST Search History

S134	5	(((instrumentation breakpoint watchpoint) with (code instruction command macro directive operation)) same (interrupt with handl$4)).clm.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/15 09:53
S135	2	((((user protected) and kernel) with (execution mode)) and (((inline insert$4 add$4) adj4 (code instruction)) with (exception near3 handler))).clm.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/24 14:09
S136	3	debug and (((code instruction) with (exception near handler)) same (preset pre-set setup set-up)) and (exception with (invok$4 forc$3))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/02/24 15:49

Terms used **code integrity anti piracy** and **exception handler same inserted inline near9 code instructions**

Found **37,892** of **197,895**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Adapting virtual machine techniques for seamless aspect support

Christoph Bockisch, Matthew Arnold, Tom Dinkelaker, Mira Mezini

October 2006 **ACM SIGPLAN Notices , Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications OOPSLA '06**, Volume 41 Issue 10

Publisher: ACM Press

Full text available: pdf(266.90 KB) Additional Information: full citation, abstract, references, index terms

Current approaches to compiling aspect-oriented programs are inefficient. This inefficiency has negative effects on the productivity of the development process and is especially prohibitive for dynamic aspect deployment. In this work, we present how well-known virtual machine techniques can be used with only slight modifications to support fast aspect deployment while retaining runtime performance. Our implementation accelerates dynamic aspect deployment by several orders of magnitude relative t ...

Keywords: aspect weaving, aspect-oriented programming, dynamic deployment, envelope-based weaving, virtual machine support

2 EDO: Exception-directed optimization in java

Takeshi Ogasawara, Hideaki Komatsu, Toshio Nakatani

January 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 1

Publisher: ACM Press

Full text available: pdf(764.07 KB) Additional Information: full citation, abstract, references, index terms

Optimizing exception handling is critical for programs that frequently throw exceptions. We observed that there are many such exception-intensive programs written in Java. There are two commonly used exception handling techniques, stack unwinding and stack cutting. Stack unwinding optimizes the normal path by leaving the exception handling path unoptimized, while stack cutting optimizes the exception handling path by adding extra work to the normal path. However, there has been no single excepti ...

Keywords: Feedback-directed dynamic optimization, dynamic compilers, exception handling, inlining

3 Measuring the dynamic behaviour of AspectJ programs

Bruno Dufour, Christopher Goard, Laurie Hendren, Oege de Moor, Ganesh Sittampalam, Clark Verbrugge
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10
Publisher: ACM Press

Full text available: pdf(226.86 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper proposes and implements a rigorous method for studying the dynamic behaviour of AspectJ programs. As part of this methodology several new metrics specific to AspectJ programs are proposed and tools for collecting the relevant metrics are presented. The major tools consist of: (1) a modified version of the AspectJ compiler that tags bytecode instructions with an indication of the cause of their generation, such as a particular feature of AspectJ; and (2) a modified version of the *J ...

Keywords: AspectJ, aspect-oriented programming, dynamic metrics, java, optimization, performance, program analysis

4 Link-time binary rewriting techniques for program compaction
Bjorn De Sutter, Bruno De Bus, Koen De Bosschere
September 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 5
Publisher: ACM Press

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Small program size is an important requirement for embedded systems with limited amounts of memory. We describe how link-time compaction through binary rewriting can achieve code size reductions of up to 62&percent; for statically bound languages such as C, C++, and Fortran, without compromising on performance. We demonstrate how the limited amount of information about a program at link time can be exploited to overcome overhead resulting from separate compilation. This is done with sc ...

Keywords: Program representation, binary rewriting, code abstraction, compaction, interprocedural analysis, linker, whole-program optimization

5 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation PLDI '00**, Volume 35 Issue 5
Publisher: ACM Press

Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

6 Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems
Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, Pradeep Khosla

October 2005 **ACM SIGOPS Operating Systems Review , Proceedings of the twentieth ACM symposium on Operating systems principles SOSP '05**, Volume 39 Issue 5
Publisher: ACM Press

Full text available: pdf(264.30 KB) Additional Information: full citation, abstract, references, citings, index terms

We propose a primitive, called Pioneer, as a first step towards verifiable code execution on untrusted legacy hosts. Pioneer does not require any hardware support such as secure co-processors or CPU-architecture extensions. We implement Pioneer on an Intel Pentium IV Xeon processor. Pioneer can be used as a basic building block to build security systems. We demonstrate this by building a kernel rootkit detector.

Keywords: dynamic root of trust, rootkit detection, self-check-summing code, software-based code attestation, verifiable code execution

7 Optimising aspectJ

Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, Julian Tibble
June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6
Publisher: ACM Press

Full text available: pdf(194.20 KB) Additional Information: full citation, abstract, references, citings, index terms

AspectJ, an aspect-oriented extension of Java, is becoming increasingly popular. However, not much work has been directed at optimising compilers for AspectJ. Optimising AOP languages provides many new and interesting challenges for compiler writers, and this paper identifies and addresses three such challenges.First, compiling *around* advice efficiently is particularly challenging. We provide a new code generation strategy for *around* advice, which (unlike previous implementations) ...

Keywords: around advice, aspect-oriented programming language, aspectJ, cflow pointcut, optimization

8 Static single assignment form for machine code

Allen Leung, Lal George
May 1999 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation PLDI '99**, Volume 34 Issue 5
Publisher: ACM Press

Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, citings, index terms

Static Single Assignment (SSA) is an effective intermediate representation in optimizing compilers. However, traditional SSA form and optimizations are not applicable to programs represented as native machine instructions because the use of dedicated registers imposed by calling conventions, the runtime system, and target architecture must be made explicit. We present a simple scheme for converting between programs in machine code and in SSA, such that references to dedicated physical registers ...

9 A region-based compilation technique for dynamic compilers

Toshio Suganuma, Toshiaki Yasue, Toshio Nakatani
January 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 1
Publisher: ACM Press

Full text available: pdf(977.63 KB) Additional Information: full citation, abstract, references, index terms

Method inlining and data flow analysis are two major optimization components for effective program transformations, but they often suffer from the existence of rarely or never executed code contained in the target method. One major problem lies in the assumption that the compilation unit is partitioned at method boundaries. This article describes the design and implementation of a region-based compilation technique in our dynamic optimization framework, in which the compiled regions are selected ...

Keywords: JIT compiler, Region-based compilation, dynamic compilation, on-stack replacement, partial inlining

10 Automated analysis: Control-flow integrity

Martín Abadi, Mihai Budiu, Úlfar Erlingsson, Jay Ligatti

November 2005 **Proceedings of the 12th ACM conference on Computer and communications security CCS '05**

Publisher: ACM Press

Full text available: pdf(218.60 KB) Additional Information: full citation, abstract, references, citings, index terms

Current software attacks often build on exploits that subvert machine-code execution. The enforcement of a basic safety property, Control-Flow Integrity (CFI), can prevent such attacks from arbitrarily controlling program behavior. CFI enforcement is simple, and its guarantees can be established formally even with respect to powerful adversaries. Moreover, CFI enforcement is practical: it is compatible with existing software and can be done efficiently using software rewriting in commodity syste ...

Keywords: binary rewriting, control-flow graph, inlined reference monitors, vulnerabilities

11 Adaptive techniques: Optimistic stack allocation for java-like languages

Erik Corry

June 2006 **Proceedings of the 2006 international symposium on Memory management ISMM '06**

Publisher: ACM Press

Full text available: pdf(155.23 KB) Additional Information: full citation, abstract, references, index terms

Stack allocation of objects offers more efficient use of cache memories on modern computers, but finding objects that can be safely stack allocated is difficult, as interprocedural escape analysis is imprecise in the presence of virtual method dispatch and dynamic class loading. We present a new technique for doing optimistic stack allocation of objects. Our technique does not require interprocedural analysis and is effective in the presence of dynamic class loading, reflection and exception han ...

Keywords: Java, garbage collection, stack allocation

12 Code management: Dynamic code management: improving whole program code locality in managed runtimes

Xianglong Huang, Brian T Lewis, Kathryn S McKinley

June 2006 **Proceedings of the second international conference on Virtual execution environments VEE '06**

Publisher: ACM Press

Full text available: pdf(153.03 KB) Additional Information: full citation, abstract, references, index terms

Poor code locality degrades application performance by increasing memory stalls due to

instruction cache and TLB misses. This problem is particularly an issue for large server applications written in languages such as Java and C# that provide just-in-time (JIT) compilation, dynamic class loading, and dynamic recompilation. However, managed runtimes also offer an opportunity to dynamically profile applications and adapt them to improve their performance. This paper describes a Dynamic Code Manage ...

Keywords: code generation, code layout, dynamic optimization, locality, performance monitoring, virtual machines

13 Inline function expansion for compiling C programs

P. P. Chang, W.-W. Hwu

June 1989 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1989 Conference on Programming language design and implementation PLDI '89**, Volume 24 Issue 7

Publisher: ACM Press

Full text available: pdf(1.14 MB) Additional Information: full citation, abstract, references, citings, index terms

Inline function expansion replaces a function call with the function body. With automatic inline function expansion, programs can be constructed with many small functions to handle complexity and then rely on the compilation to eliminate most of the function calls. Therefore, inline expansion serves a tool for satisfying two conflicting goals: minizing the complexity of the program development and minimizing the function call overhead of program execution. A simple inline expansion procedur ...

14 Dynamic compilation techniques: Optimized interval splitting in a linear scan register allocator

Christian Wimmer, Hanspeter Mössenböck

June 2005 **Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments VEE '05**

Publisher: ACM Press

Full text available: pdf(341.54 KB) Additional Information: full citation, abstract, references, citings, index terms

We present an optimized implementation of the linear scan register allocation algorithm for Sun Microsystems' Java HotSpot™ client compiler. Linear scan register allocation is especially suitable for just-in-time compilers because it is faster than the common graph-coloring approach and yields results of nearly the same quality.Our allocator improves the basic linear scan algorithm by adding more advanced optimizations: It makes use of lifetime holes, splits intervals if the register press ...

Keywords: compilers, graph-coloring, java, just-in-time compilation, linear scan, optimization, register allocation

15 A study of exception handling and its dynamic optimization in Java

Takeshi Ogasawara, Hideaki Komatsu, Toshio Nakatani

October 2001 **ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications OOPSLA '01**, Volume 36 Issue 11

Publisher: ACM Press

Full text available: pdf(190.18 KB) Additional Information: full citation, abstract, references, citings, index terms

Optimizing exception handling is critical for programs that frequently throw exceptions. We observed that there are many such exception-intensive programs iin various categories of Java programs. There are two commonly used exception handling techniques, stack

unwinding optimizes the normal path, while stack cutting optimizes the exception handling path. However, there has been no single exception handling technique to optimize both paths.

16 Tdb: a source-level debugger for dynamically translated programs

Naveen Kumar, Bruce R. Childers, Mary Lou Soffa
September 2005 **Proceedings of the sixth international symposium on Automated analysis-driven debugging AADEBUG'05**
Publisher: ACM Press

Full text available: pdf(158.26 KB) Additional Information: full citation, abstract, references, citings, index terms

Debugging techniques have evolved over the years in response to changes in programming languages, implementation techniques, and user needs. A new type of implementation vehicle for software has emerged that, once again, requires new debugging techniques. Software dynamic translation (SDT) has received much attention due to compelling applications of the technology, including software security checking, binary translation, and dynamic optimization. Using SDT, program code changes dynamically, an ...

Keywords: debugging, dynamic binary translation, dynamic instrumentation

17 Efficient Java exception handling in just-in-time compilation

Seungil Lee, Byung-Sun Yang, Suhyun Kim, Seongbae Park, Soo-Mook Moon, Kemal Ebcioğlu, Erik Altman
June 2000 **Proceedings of the ACM 2000 conference on Java Grande JAVA '00**
Publisher: ACM Press
Full text available: pdf(641.73 KB) Additional Information: full citation, references, citings, index terms

18 Architectural support for software-based protection

Mihai Budiu, Úlfar Erlingsson, Martín Abadi
October 2006 **Proceedings of the 1st workshop on Architectural and system support for improving software dependability ASID '06**
Publisher: ACM Press
Full text available: pdf(642.62 KB) Additional Information: full citation, abstract, references, index terms

Control-Flow Integrity (CFI) is a property that guarantees program control flow cannot be subverted by a malicious adversary, even if the adversary has complete control of data memory. We have shown in prior work how CFI can be enforced by using inlined software guards that perform safety checks. The first part of this paper shows how modest Instruction Set Architecture (ISA) support can replace such guard code with single instructions.On the foundation of CFI we have implemented XFI: a protecti ...

Keywords: binary rewriting, control-flow graph, control-flow integrity, hardware support, memory protection, security, software fault isolation

19 Effectiveness of cross-platform optimizations for a java just-in-time compiler

Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Gohda, Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue, Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11

Publisher: ACM Press

Full text available: pdf(405.65 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes the system overview of our Java Just-In-Time (JIT) compiler, which is the basis for the latest production version of IBM Java JIT compiler that supports a diversity of processor architectures including both 32-bit and 64-bit modes, CISC, RISC, and VLIW architectures. In particular, we focus on the design and evaluation of the cross-platform optimizations that are common across different architectures. We studied the effectiveness of each optimization by selectively disabling ...

Keywords: Java, just-in-time compiler, optimization

20 Session 3: sampling: Low-overhead call path profiling of unmodified, optimized code

Nathan Froyd, John Mellor-Crummey, Rob Fowler

June 2005 **Proceedings of the 19th annual international conference on Supercomputing ICS '05**

Publisher: ACM Press

Full text available: pdf(399.57 KB) Additional Information: full citation, abstract, references, citings

Call path profiling associates resource consumption with the calling context in which resources were consumed. We describe the design and implementation of a low-overhead call path profiler based on stack sampling. The profiler uses a novel sample-driven strategy for collecting frequency counts for call graph edges without instrumenting every procedure's code to count them. The data structures and algorithms used are efficient enough to construct the complete calling context tree exposed during ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Web Results **1 - 10** of about **169** for **kernel user mode anti-piracy embedding code inline instruction force**

LWN.net weekly edition
This **kernel** now has over 90 patches; some of the latest additions are TUX, **User-mode** Linux, the Linux Trace toolkit, and more. Core **kernel code**: ...
lwn.net/2002/0328/bigpage.php3 - 139k - Cached - Similar pages

[PDF] MAI Thesis
File Format: PDF/Adobe Acrobat - View as HTML
distinguish between **user-mode** and **kernel-mode** debuggers. **User-mode** debuggers ... This **instruction** calls the debug **exception handler**, which allows ...
https://www.cosic.esat.kuleuven.be/publications/thesis-122.pdf - Similar pages

2005 Newsgroup postings (4/22 - 5/4) Anne & Lynn Wheeler
There is one **exception**; where a CA distributes its public key in the form of a ... most vendors got around to rewriting their FINWAIT list **handling code**. ...
www.garlic.com/~lynn/2005g.html - 202k - Cached - Similar pages

Bistouri news aggregator
Since the goal is to stage **user-mode** payloads from **kernel-mode**, the overriden ... (d78.33c): Break **instruction exception - code** 80000003 (first chance) ...
dormrf.free.fr/rss/fr/?hours=168 - 989k - Cached - Similar pages

LXer: Release-critical Bugreport for November 3, 2006
Excellent news for **embedded** Linux. See also their sister site DesktopLinux.com. ... **Force** 383465 [I] Contains obfuscated source **code**, DFSG violation? ...
lxer.com/module/newswire/view/73506/index.html - 132k - Cached - Similar pages

Gregarius » OSNews » January 2007
"**User Mode** Linux allows you to run Linux kernels as **user mode** processes under a host Linux **kernel**, giving you a simple way to run several independent ...
rss.vacantmind.net/OSNews/2007/01/ - 317k - Cached - Similar pages

[DOC] MSIN60
File Format: Microsoft Word - View as HTML
The **exception** is Windows Server 2003, Web Edition, on which IIS is installed ... There are two types of errors that are reported, **user mode** and **kernel mode**. ...
download.microsoft.com/download/f/d/a/fda0c216-9bc2-4d67-90f3-3855cf500021/Mn_Srv03.doc - Similar pages

[DOC] Using Windows Server 2003 with Service Pack 1 in a Managed ...
File Format: Microsoft Word - View as HTML
Product activation is an **anti-piracy** technology designed to verify that software products ... When a **user mode** error occurs, such as an application error, ...
download.microsoft.com/download/f/d/a/fda0c216-9bc2-4d67-90f3-3855cf500021/Mn_Srv03SP1.doc - Similar pages

jkOnTheRun: pocket pc
Strict partitioning of **user mode** and **kernel mode** (no SetKMode! ... an update to the Pocket Mechanic program that includes some malicious **anti-piracy code**. ...
jkontherun.blogs.com/jkontherun/pocket_pc/index.html - 824k - Cached - Similar pages

Hardware reviews, Overclocking, Tech News and Enthusiast Community ...
"Apple hasn't fixed any of the bugs published during the Month of **Kernel** Bugs, except for the AirPort issue," said "LMH," the **code** name of the security ...
www.overclockersclub.com/?read=6360189 - 83k - Cached - Similar pages

Result Page: 1 2 3 4 5 6 7 8 9 10 **Next**
